

Podstawy Programowania

Laboratorium 2

Standardowe algorytmy iteracyjne

prowadzący: mgr inż. Marta Lampasiak, mgr inż. Michał Jaroszczuk

1 Wprowadzenie

Celem zajęć jest poznanie podstaw programowania w języku C/C++, w szczególności ćwiczenia z tworzeniem programów ilustrujących zastosowanie instrukcji pętlowych (**while**, **do while**, **for**). Standardowe algorytmy iteracyjne: zliczanie, sumowanie, szukanie maksimum i minimum. Ćwiczenia z tworzeniem własnych funkcji. Funkcje bezparametrowe i zmienne lokalne. Przekazywanie parametrów przez zmienne globalne.

Podczas programowania zawsze spotykamy się z operacjami, które należy wykonywać wielokrotnie. W celu powtórzenia danego ciągu instrukcji należy skorzystać z pętli. Istnieją trzy typy pętli. Każda pętla składa się zawsze z góry określonego warunku końca oraz powtarzanych instrukcji. Poniżej pokazano w jaki sposób można wypisać ciąg liczb od 1 do 9 włącznie przy użyciu różnych pętli. Jak widać pętle można stosować zamiennie.

```
1 #include <iostream>
2
3 int main()
4 {
5     for (int a = 1; a < 10; a++)
6         std::cout << a << " ";
7     std::cout << "\n";
8
9     int b = 1;
10    while (b < 10)
11        std::cout << b++ << " ";
12    std::cout << "\n";
13
14    int c = 1;
15    do
16        std::cout << c++ << " ";
17    while (c < 10);
18    std::cout << "\n";
19 }
```

Jednak każdą pętlę charakteryzuje inna cecha/zastosowanie. Pętli **for** używa się najczęściej w przypadku kiedy pętla ma się wykonać z góry określoną liczbę razy (dokładnie tak jak w pokazanym przypadku). W przeciwnym razie, kiedy nie wiemy ile dokładnie razy ma wykonać się pętla (np.: aż do podania konkretniej wartości), korzystamy z pętli **while** lub **do while**. Jeśli chcemy zagwarantować, aby pętla wykonała się chociaż raz to należy użyć **do while**, ponieważ warunek jest sprawdziany na końcu.

W celu uporządkowania oraz poprawienia czytelności programu częstym zabiegiem jest wydzielenia fragmentów programu do funkcji. W szczególności wydzielanymi fragmentami są instrukcje, które chcemy powtarzać. Należy pamiętać, że podobnie jak w przypadku definiowania zmiennych należy zawsze określić typ funkcji (**void**, **int**, **bool** itp.). Ponadto cechą charakterystyczną dla funkcji są nawiasy "(" , ")" umieszczone bezpośrednio po nazwie funkcji. Ciało funkcji (instrukcje wewnątrz funkcji) są zgrupowane wewnątrz nawiasów "{" , "}". Poprzez polecenie **return** można wyjść z funkcji wraz z określoną wartością. Wartość zwracana przez funkcję musi być zgodna z jej typem! W jednej funkcji polecenie **return** może pojawiać się więcej niż jeden raz w zależności od rozgałęzienia funkcji warunkowych.

```

1 #include <iostream>
2
3 int a;
4
5 bool odd() {
6     return a % 2;
7 }
8
9 int main()
10 {
11     a = 1;
12     for (; a < 10; a++) {
13         if (odd())
14             std::cout <<a<< " - nieparzysta\n";
15         else
16             std::cout <<a<< " - parzysta\n";
17     }
18 }

```

Powyżej przedstawiono kompletny kod programu, który ma zadanie określić parzystość liczb całkowitych z zakresu $[1, 10)$. Proszę zwrócić uwagę, że pole inicjalizacyjne w pętli `for` zostało nieuzupełnione. Pole może pozostać puste, jednak trzeba pamiętać o wszystkich średnikach.

2 Zadania

1. OCENA 3:

Napisz program z wykorzystaniem funkcji, która będzie sumować wprowadzane przez użytkownika kolejne liczby do momentu podania liczby 99. Ponadto należy sumować liczby wyłącznie z zakresu od -15 do 15 włącznie. Należy przechowywać jedynie ostatnią liczbę. Jeśli użytkownik wprowadzi następujący ciąg liczb: 20 -7 -30 2 15 3 99 to suma wczytanych liczb wynosi: 13.

2. OCENA 4:

Napisz menu użytkownika, które pozwoli na wybór programu z zadania 1 oraz obydwu programów z zadania 2. Menu powinno działać w pętli i pozwalać na wielokrotne włączanie podprogramów. Program powinien kończyć pracę poprzez wybór odpowiedniej opcji. Do wywołania poszczególnych programów wykorzystaj funkcję typu *void*.

- (a) Napisz program do obliczania największego wspólnego dzielnika wprowadzonych dwóch liczb. Należy zastosować algorytm Euklidesa i przyjąć: $a, b \in C_+$

Algorithm 1: algorytm Euklidesa

```

1 function NWD( $a, b$ ):
2     while  $a \neq b$  do
3         if  $a > b$  then
4              $a \leftarrow a - b$ ;
5         else
6              $b \leftarrow b - a$ ;
7     return  $a$ ;

```
